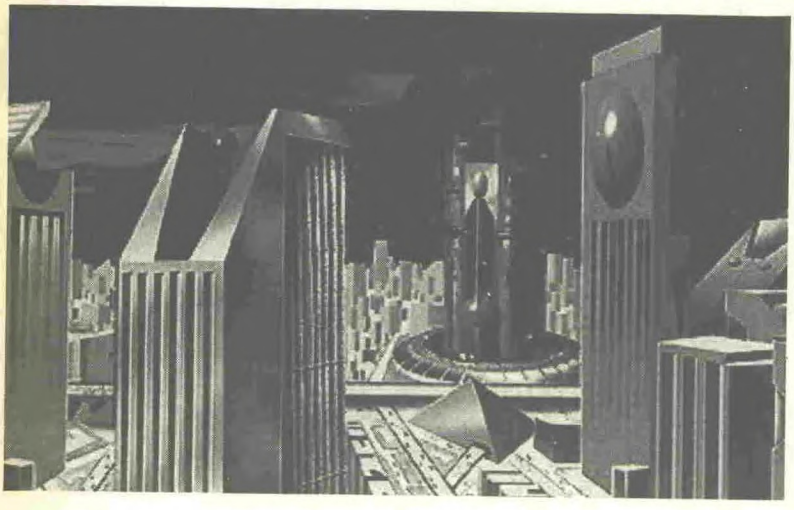
การสร้างภาพ 3 มิติ ใน 5 ระบบ



แนะนำระบบต่าง ๆ ในการสร้างภาพ 3 มิติ เพื่อจะได้ทราบถึงขอบเขต ทั้งหมดอย่างถูกต้อง

เท่าที่ได้ติดตามบทความด้านกราฟิก ในวารสารทางคอมพิวเตอร์ต่าง ๆ ระยะนานพอควร ทำให้ทราบว่าผู้เขียนบทความหลาย ๆ ท่านได้เสนอหลักการสร้างภาพ 3 มิติในแบบเดียวหรือระบบเดียวกันหมด เนื่องจากเป็นแบบเหมาะสำหรับการเริ่มต้นเรียนรู้ โดยไม่ได้กล่าวถึงระบบอื่น ๆ ไว้ ซึ่งอาจทำให้ผู้สนใจการสร้างภาพ 3 มิติ ทราบเนื้อหาไม่ครบ เมื่อต้องการพัฒนางานบางอย่างอาจพบปัญหา

จริง ๆ แล้วการสร้างภาพ 3 มิติ ยังมีอีกหลายระบบหรือหลายวิธีการ ซึ่งจะกล่าวแนะนำระบบต่าง ๆ ทั้งหมดในบทความนี้ เพื่อให้ผู้สนใจทางกราฟิกได้ทราบขอบเขตทั้งหมดอย่างถูกต้อง เพื่อค้นคว้าเพิ่มเติมและเพื่อการพัฒนาโปรแกรมทางกราฟิกได้ดียิ่งขึ้น

ตามวิธีการของ ALAN WATT ได้บ่งระบบการมอง (Viewing system) หรือระบบการสร้างภาพออกเป็น 5 ระบบ เรียงลำดับจากระบบง่าย ๆ ไปยาก จากระบบเบื้องต้นจนถึงระบบซับซ้อน จนถึงระบบมาตรฐานกราฟิก คือ Graphical Kernel System (GKS) หรือ Programmer's Hier archical Graphics System (PHIGS) บทความนี้จะกล่าวเป็นเชิงแนะนำเท่านั้น เพราะถ้าอธิบายโดยละเอียดแล้วคงต้องเข้าห้องเรียน เพื่อฟังการบรรยายกันหลายวัน (หมายถึงเข้าห้องปฏิบัติการด้วยนะครับ) วิธีของ ALAN WATT ได้บ่งระบบออกเป็นดังนี้

Viewing system I

มีลักษณะ

1. เป็นระบบง่ายสุดการเริ่มต้นต้องการ

(a) จุดผู้มอง C (camera point) ใน 3 มิติ 1 จุด (กำหนดพิกัดในระบบทรงกลม) ใน World Coordinate, WC

(b) ระยะห่างจากจุดผู้มองไปฉากรับภาพ (view plane) กำหนดเป็น d

2. ทิศทางการมองจำกัด คือต้องมองออกจากจุดผู้มองไปยังจุดกำเนิดได้เท่านั้น จะกำหนดมองไปในทิศอื่น ๆ หรือจุดอื่นไม่ได้

3. ทิศทางการหันฉากรับภาพจำกัด คือต้องตั้งฉากกับระยะจุดผู้มองไปจุดกำเนิดใน WC เท่านั้น

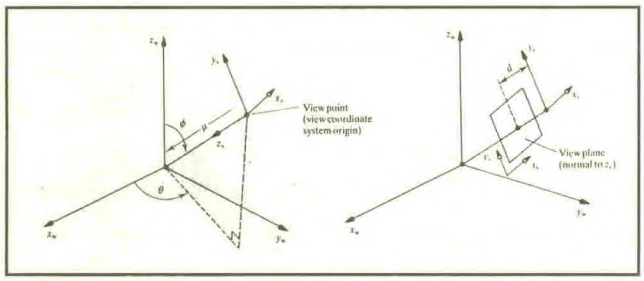
4. ไม่ใช้ view volumn (เริ่มใช้ในระบบ IV)

5. ไม่ใช้วินโดว์บนฉากรับภาพ (วินโดว์ในที่นี้หมายถึงช่องสี่เหลี่ยมเท่านั้น ท่านผู้อ่านอย่าสับสนกับวินโดวส์ 3.11 หรือวินโดวส์ 95)

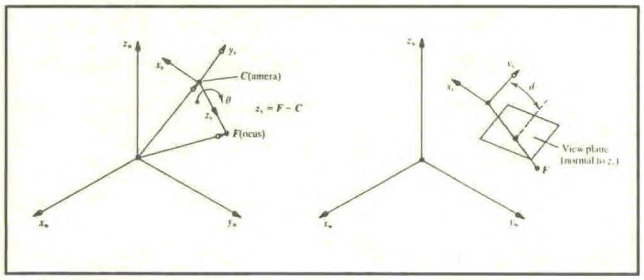
6. จุดศูนย์กลางฉายภาพ (center of projection) เป็นจุดเดียวกับจุดผู้มอง ทำในแบบเปอร์สเปกทีฟ (perspective)

ระบบนี้ใช้ได้ดีกับวัตถุจำนวนไม่มาก และมีพิกัดอยู่ในบริเวณจุดกำเนิดใน WC ถ้าหากวัตถุอยู่ไกลไปจากจุดกำเนิด ต้องย่อภาพทั้งหมดให้เล็กลงเพื่อจะได้เห็นภาพบนฉากรับภาพ เราสามารถหมุนภาพไปมาได้ด้วยการปรับมุมสองมุมในระบบพิกัดทรงกลม

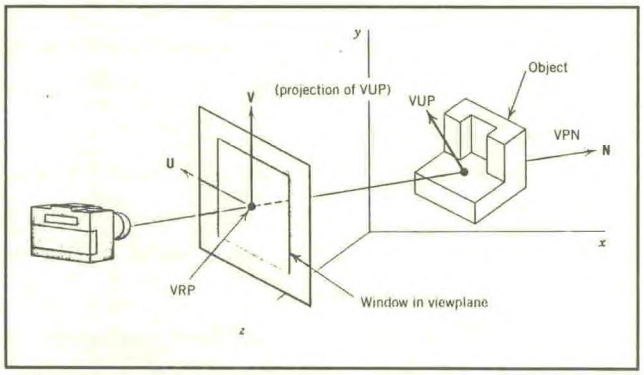
ระบบนี้เหมาะกับการเริ่มต้นเรียนรู้ทำความเข้าใจ ซึ่งผู้เขียนบทความการสร้างภาพ 3 มิติ หลายท่านจะใช้ระบบนี้เป็นหลัก แต่ไม่เหมาะกับจำนวนวัตถุมาก ๆ ที่ไกลไปจากจุดกำเนิด การหมุนภาพเพื่อดูในมุมต่าง ๆ ทำได้เฉพาะจุดกำเนิดเท่านั้น หากต้องการหมุนภาพ รอบจุดอื่น ๆ ที่ไม่ใช่จุดกำเนิดจะทำไม่ได้ นับเป็นข้อจำกัดอย่างมากของระบบนี้ ดูรูปที่ 1



รูปที่ 1 viewing system I, d เป็นระยะห่างฉากรับภาพ



รูปที่ 2 Viewing system II



รูปที่ 3 Viewing system III จุด VRP ในรูปคือจุด C

Viewing system II

มีลักษณะ

1. ระบบต้องการ

(a) จุดผู้มองและจุดเป้าหมาย (focus) ใน WC

(b) มุมหมุน (twist angle) เพื่อผู้มองปรับมุมการมอง (สามารถเอียงศีรษะไปมาได้)

(c) ระยะห่างฉากรับภาพ

2. ไม่ใช้ view volumn

3. ไม่ใช้วินโดว์บนฉากรับภาพ

4. จุดศูนย์กลางฉายภาพเป็นจุดเดียวกับผู้มอง ทำในแบบเปอร์สเปกทีฟ จะเห็นได้ว่าระบบนี้การมองไปยังจุดต่าง ๆ เป็นอิสระมากขึ้น ไม่ถูกจำกัดให้มองเฉพาะไปที่จุดกำเนิดเพียงอย่างเดียวดังเช่นระบบแรก ดูรูปที่ 2

Viewing system III

มีลักษณะ

1. ระบบต้องการ

(a) จุดผู้มองและเวกเตอร์สองตัวคือ เวกเตอร์การแสดงทิศทางการมอง N ซึ่งจะตั้งฉากกับฉากรับภาพ และเวกเตอร์ชี้ขึ้น Vup

(b) ระยะห่างฉากรับภาพ

2. ไม่ใช้ view volumn

3. ไม่ใช้วินโดว์บนฉากรับภาพ

4. จุดศูนย์กลางฉายภาพเป็นจุดเดียวกับผู้มอง ทำในแบบเปอร์สเปกทีฟ เป็นระบบที่เริ่มใกล้เคียงสู่ระบบมาตรฐาน GKS, PHIGS จะมองไปในทิศใดก็ได้เช่นเดียวกับระบบ II การคำนวณเริ่มยุ่งยากขึ้น เริ่มจากหาเวกเตอร์ตัวที่สาม คือ U ซึ่งจะต้องตั้งฉากกับ N และ Vup หลังจากนั้นจะย้อนกลับไปหาเวกเตอร์ชี้ขึ้นตัวใหม่ V ที่จะต้องตั้งฉาก ทั้ง N และ U อีกครั้ง ทั้งนี้เนื่องจาก Vup ที่เรากำหนดขึ้นนั้น ยังไม่ใช้เวกเตอร์ชี้ขึ้นจริงที่ใช้ในการคำนวณ แต่กำหนดขึ้นตามความรู้สึกของผู้มอง ระบบนี้ยังไม่ใช้ view volumn แต่จะเป็นพื้นฐานไปสู่ระบบต่อไป (ดูรูปที่ 3)

Viewing system IV

มีลักษณะ

1. ระบบต้องการ

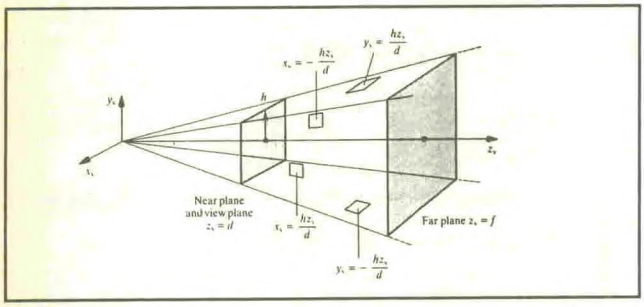
(a) จุดผู้มองและเวกเตอร์สองตัวคือเวกเตอร์การแสดงทิศทางการมอง N ซึ่งจะตั้งฉากกับฉากรับภาพและเวกเตอร์ชี้ขึ้น V เช่นเดียวกับระบบ III

(b) ระยะห่างระนาบด้านหน้า (near plane distance) และระยะห่างระนาบด้านหลัง (far plane distance)

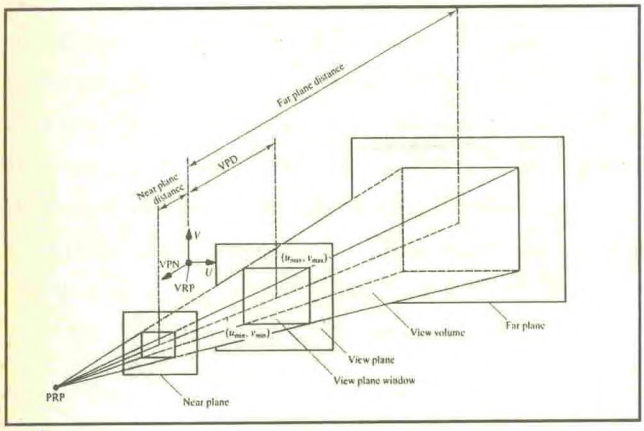
2. มีการใช้ view volumn แต่กำหนดให้ฉากรับภาพอยู่ที่เดียวกับระนาบด้านหน้า

3. มีการใช้วินโดว์บนฉากรับภาพ แต่ยังเป็นแบบจตุรัส

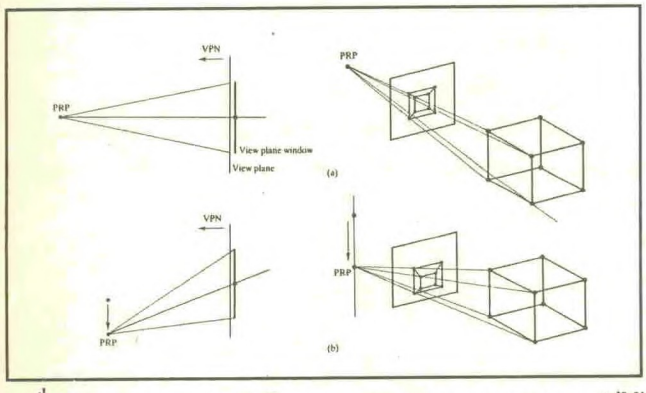
4. จุดศูนย์กลางฉายภาพเป็นจุดเดียวกับผู้มอง ทำในแบบเปอร์สเปกทีฟ ในระบบนี้เริ่มมีการใช้ view volumn ซึ่งมีลักษณะเหมือนปิรามิดที่ถูกตัดยอด (frustum) สิ่งใดอยู่ใน view volumn จึงจะปรากฏบนฉากรับภาพที่กำหนดขนาดเป็นวินโดว์ สิ่งใดอยู่นอก view column สิ่งนั้นจะถูกตัด (clip) ออกไปซึ่งจะตรงกับความต้องการมากสุด ลักษณะเช่นนี้จะต่างไปจากสามระบบแรกที่กล่าวมา คือ ในสามระบบแรกจะรับทุกอย่างที่ได้จากการฉาย (project) ลงบนฉาก แม้แต่สิ่งที่อยู่ระหว่างระนาบด้านหน้าไปยังจุดผู้มอง หรือสิ่งที่อยู่หลังจากระนาบด้านหลังห่างออกไปอย่างไม่จำกัด ซึ่งสิ่งต่าง ๆ เหล่านี้เราไม่ต้องการ เพราะมันจะทำให้ยากต่อการคำนวณ เพื่อสร้างภาพให้ใกล้เคียงความจริง (render) เมื่อมีการให้แสง สี เงา แก่ผิววัตถุทั้งหลายในฉาก



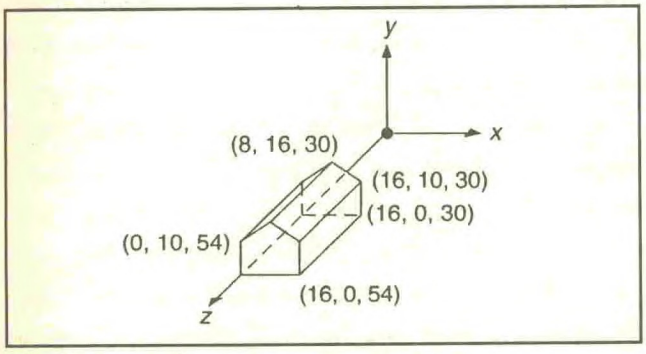
รูปที่ 4 Viewing system IV



รูปที่ 5 System V



รูปที่ 6 (a) แสดงการฉายปกติ (b) แสดงการฉายแบบ Oblique ทำได้โดยการเลื่อนจุด PRP ลงมาในแนวดิ่ง



รูป 7 บ้านมีพิกัดอยู่ใน World-Coordinate (WC)

เนื่องจากในการคำนวณทางกราฟิก เราจะเปรียบเทียบความลึก คือค่าทางแกน z (พิกัดวัดจากจุดมอง) ระหว่างผิววัตถุทั้งหลาย เพื่อทำการซ่อน (hidden) กำหนดระดับความเข้มสีแก่ผิวที่จะมองเห็น วัตถุใดมีผิวลึกสุดจึงหมายถึงผิวที่อยู่ติดกับระนาบด้านหลังของ view volumn ซึ่งผิวนี้จะถูกบังเสมอ หากพบผิววัตถุอื่นใดอยู่หน้ามัน วัตถุใดอยู่เลยระนาบด้านหลังของ view volumn ออกไป เราไม่ต้องคำนวณให้เสียเวลา

ทำนองเดียวกันวัตถุใดมีผิวอยู่หน้าสุด คือผิวที่อยู่ติดกับระนาบด้านหน้าของ view volumn ผิวนี้จะบังทุกผิวบนฉากนั้น ๆ (ถ้าชนิดของผิววัตถุเป็นแบบทึบ ไม่โปร่งใส) ไม่มีผิววัตถุใดอยู่หน้าไปกว่านี้อีกแล้ว ทำให้การคำนวณให้สีผิววัตถุ (ตามข้อกำหนดต่าง ๆ) สามารถตัดสินใจได้ง่ายและเร็วขึ้น จะเห็นได้ว่าในสามระบบแรกเหมาะกับการสร้างภาพในแบบโครง (wire frame) เท่านั้น (ดูรูปที่ 4)

Viewing system V

ระบบนี้เป็นไปตามมาตรฐาน ของ GKS, PHIGS มีลักษณะ

1. ระบบต้องการ

(a) จุดผู้มองและเวกเตอร์สองตัวคือเวกเตอร์การแสดงทิศทางการมอง VPN ซึ่งจะตั้งฉากกับฉากรับภาพและเวกเตอร์ชี้ขึ้น VUV เช่นเดียวกับระบบ IV

(b) ระยะห่างระนาบด้านหน้า ระยะห่างระนาบด้านหลัง และระยะฉากรับภาพ

2. มีการใช้ view volumn ฉากรับภาพจะอยู่ที่ใดก็ได้ไม่จำเป็นต้องอยู่ระนาบด้านหน้าเช่นเดียวกับระบบ IV

3. มีการใช้วินโดวส์บนฉากรับภาพ เป็นสี่เหลี่ยมผืนผ้าแบบใดก็ได้ ไม่จำเป็นต้องเป็นจตุรัส กำหนดโดยตัวแปร 4 ตัว คือ Umax, Vmax, Umin, Vmin

4. จุดศูนย์กลางฉายภาพกับจุดผู้มอง ไม่จำเป็นต้องอยู่ที่เดียวกัน

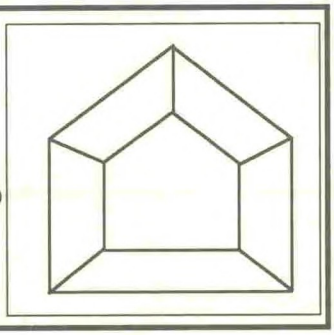
5. ทำการฉาย ในแบบเปอร์สเปกทีฟ (ทำได้มากกว่า 1 จุด)

6. ทำการฉายในแบบ Oblique หรือชนิด Orthograhics ได้

ในระบบนี้ค่อนข้างต่างไปจากระบบก่อนมาก มีตัวแปรต่าง ๆ เข้ามาเกี่ยวข้องมากขึ้น มีความซับซ้อนมากขึ้น ที่สังเกตได้ชัดคือจุดผู้มอง (ใน GKS เรียกจุด VRP) อยู่ยกออกมาจากจุดศูนย์กลางฉายภาพ (ใน GKS เรียก PRP) ทำให้สร้างภาพลงบนฉากรับภาพได้หลายแบบ ดูรูป 6

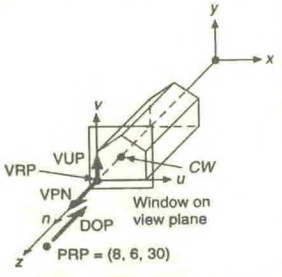
ผู้ใช้ต้องมีความเข้าใจลึกซึ้งในการตั้งค่าตัวแปรต่าง ๆ เช่น ตำหน่งของจุด VRP จุดศูนย์กลางฉายภาพ PRP ทิศทางการมอง VPN ระยะห่างระนาบด้านหน้า ระยะห่างระนาบด้านหลัง ระยะห่างฉากรับภาพ การตั้งค่ากรอบวินโดว์ เป็นต้น ดูตัวอย่างต่อไปนี้ รูป 7 เป็นภาพบ้านต้นแบบ

VRP(WC)	(0, 0, 0)
VPN(WC)	(0, 0, 1)
VUP(WC)	(0, 1, 0)
PRP(VRC)	(8, 6, 84)
window(VRC)	(-50, 50, -50, 50)
projection type	perspective



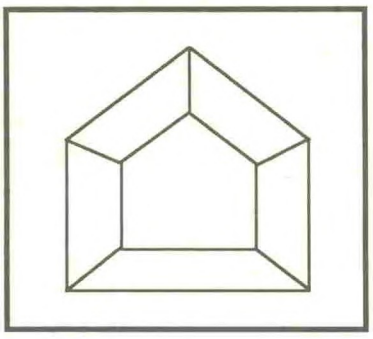
รูป 8 ผลที่ได้

VRP(WC)	(0, 0, 54)
VPN(WC)	(0, 0, 1)
VUP(WC)	(0, 1, 0)
PRP(VRC)	(8, 6, 30)
window(VRC)	(-1, 17, -1, 17)
projection type	perspective

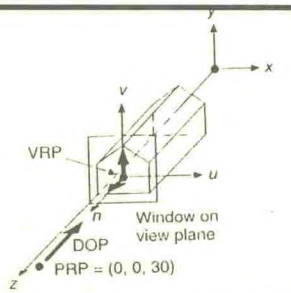


รูปที่ 9 แสดงตำแหน่งต่าง ๆ

รูปที่ 10
บ้านอยู่กลางวินโดว์

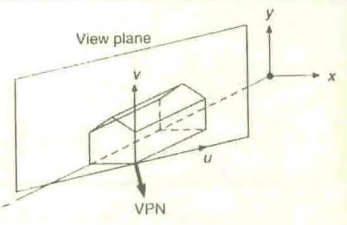


VRP(WC)	(8, 6, 54)
VPN(WC)	(0, 0, 1)
VUP(WC)	(0, 1, 0)
PRP(VRC)	(0, 0, 30)
window(VRC)	(-9, 9, -7, 11)
projection type	perspective

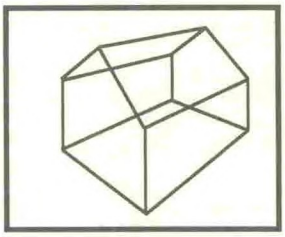


รูปที่ 11 แสดงตำแหน่งต่าง ๆ ผลที่ได้เหมือนรูปที่ 10

VRP(WC)	(16, 0, 54)
VPN(WC)	(1, 0, 1)
VUP(WC)	(0, 1, 0)
PRP(VRC)	(0, 25, $20\sqrt{2}$)
window(VRC)	(-20, 20, -5, 35)
projection type	perspective



รูปที่ 12 การตั้งค่าแบบ 2 จุดเปอร์สเปกทีฟ



รูปที่ 13
ผลที่ได้

เมื่อกำหนดค่าต่าง ๆ ดังนี้ ผู้มอง VRP(WC) อยู่ที่จุดกำเนิด มีพิกัดเป็น (0, 0, 0) มองไปตามแกน z ได้ VPN(WC) เป็น (0, 0, 1) เวกเตอร์ชี้ขึ้น VUP (WC) ชี้ปตามแกน y ได้เป็น (0, 1, 0) จุดศูนย์กลางรับภาพ PRP(VRC) วัดพิกัดเทียบกับแกน VRC (อยู่ที่จุดผู้มอง) ได้ (8, 6, 84) กำหนดขนาดวินโดวส์ (-50, 50, -50, 50) เรากำหนดให้มากเพื่อจะได้มองเห็นวัตถุได้ในแนวกว้าง ฉากรับภาพอยู่ที่ผู้มองและเลือกการฉายในแบบเปอร์สเปกทีฟ จะได้ตามรูป 8

จากรูปที่ 8 ภาพบ้านที่ได้เป็นเปอร์สเปกทีฟขนาดเล็กไม่อยู่ตรงกลางวินโดว์ จะปรับใหม่เลื่อนผู้มองและฉากรับภาพมาอยู่ด้านหน้าคือ VRP(WC) อยู่ที่ (0, 0, 54) จุด PRP ปรับใหม่เมื่อเทียบกับ VRC ต่อสรุปว่าอยู่ที่เดิมคือ (8, 6, 30) และเนื่องจากบ้านกว้าง 16 หน่วย เราจะตั้งกรอบวินโดว์เป็น (-1, 17, -1, 17) จะได้ตามรูปที่ 9 ผลตามรูปที่ 10

จากรูป 10 เราสามารถกำหนดในแบบอื่นได้อีกดังรูปที่ 11

ที่ผ่านมาเป็นการทำบบ 1 จุดเปอร์สเปกทีฟ ต่อไปนี้เป็นการตั้งแบบ 2 จุดเปอร์สเปกทีฟตั้งค่าตามรูป 12 ได้ผลตามรูป 13

จะเห็นได้ว่าการตั้งค่าต่าง ๆ เหล่านี้ต้องมีความสัมพันธ์กัน หากไม่เช่นนั้นแล้ว เราจะไม่เห็นบ้านปรากฏบนจอ หรืออาจควบคุมขนาดบ้านที่ปรากฏบนจอให้ตรงกับความต้องการไม่ได้เป็นต้น

ตัวอย่างโปรแกรม viewing system I

ในระบบ I นี้ ค่า xv,yv,zv ที่คำนวณได้จากแมตริกซ์ของการแปลงดังนี้

$$[X_v\ Y_v\ Z_v\ 1] = [X_w\ Y_w\ Z_w\ 1] \begin{bmatrix} -\text{SIN}\theta & -\text{COS}\theta\text{COS}\phi & -\text{COS}\theta\text{SIN}\phi & 0 \\ \text{COS}\theta & -\text{SIN}\theta\text{COS}\phi & -\text{SIN}\theta\text{SIN}\phi & 0 \\ 0 & \text{SIN}\theta & -\text{COS}\phi & 0 \\ 0 & 0 & \mu & 1 \end{bmatrix}$$

$$= [X_w\ Y_w\ Z_w\ 1]\ T_{view}$$

ซึ่งถ่ายทอดเป็นโปรแกรมได้ตามตัวอย่างโปรแกรม 1 พร้อมแสดงภาพเคลื่อนไหวด้วยการ swap page

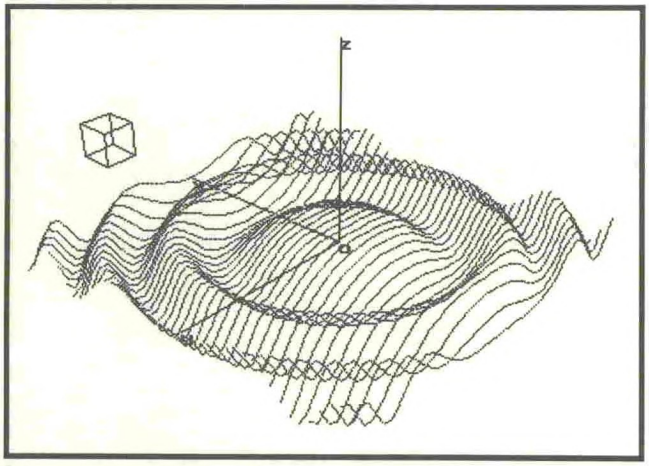
โปรแกรมที่ 1

```
program Example_1_Draw_surface_wit h_Viewing_system_1;
{== Somchart R.SORAKARN,
     Dept. of Comp. Science,
       KMITNB, BKK 10800.===}
uses crt,graph;
const  D=600;    rho=50;
var   ch:char;
     s1,s2,c1,c2,Th,Ph:real;
         i,Ipage,Xmax,Ymax,sx,sy:integer;

procedure Per(L:char;x,y,z:real);
var    xe,ye,ze:real;
begin  xe:=-x*s1+ y*c1;                          {ในที่นี้ Xe=Xv, Ye=Yv และ Ze=Zv}
      ye:=-x*c1*c2 - y*s1*c2 + z*s2;
      ze:=-x*s2*c1 - y*s1*s2 - z*c2 + rho;
      sx:=trunc(Xmax+D*xe/ze);
      sy:=trunc(Ymax-D*ye/ze);
      if upcase(L)='L' then LineTo(sx,sy);
      if Upcase(L)='M' then MoveTo(sx,sy);
end;
procedure XYZ;
var Ox,Oy:integer;
   t,p:real;
begin
    t:=3.14159/180*Th;                           {ทำองศาเป็นมุมเรเดียน}
    p:=3.14159/180*Ph;
    setcolor(15);
    s1:=sin(t);  c1:=cos(t);
    s2:=sin(p);  c2:=cos(p);
    per('M',0,0,0);   OutTextXY(sx,sy,'o');Ox:=sx;Oy:=sy;
    per('L',0,0,10.2); OutTextXY(sx,sy,'z');MoveTo(Ox,Oy);
    per('L',0,10.2,0); OutTextXY(sx,sy,'y');MoveTo(Ox,Oy);
    per('L',10.2,0,0); OutTextXY(sx,sy,'x');
end;

procedure Init_Graph;
var gd,gm:integer;
begin
   gd:=9; gm:=1;
   initGraph(gd,gm,'');
   XYZ;
end;

procedure curve;
var Fl:integer; x,y,z:real;
begin
  x:=-10; setcolor(11);
  repeat
    x:=x+0.5; Fl:=0; y:=-10;
   repeat
    y:=y+0.5;
     z:=cos(0.1*(x*x+y*y));
     if (Fl=0) then begin per('M',x,y,z);Fl:=1 end else per('L',x,y,z);
   until y>10;
  until  x>10;
end;
```

→

โปรแกรมที่ 1 (ต่อ)

```
procedure Cube(xo,yo,zo,x1,y1,z1,co:integer);
begin
  setcolor(co);
  per('M',x1,yo,zo);per('L',x1,y1,zo);per('L',x1,y1,z1);
                per('L',x1,yo,z1);per('L',x1,yo,zo);
  per('M',xo,yo,zo);per('L',xo,y1,zo);per('L',xo,y1,z1);
                per('L',xo,yo,z1);per('L',xo,yo,zo);
  per('M',xo,y1,zo);per('L',x1,y1,zo);
  per('M',xo,y1,z1);per('L',x1,y1,z1);
  per('M',xo,yo,zo);per('L',x1,yo,zo);
  per('M',xo,yo,z1);per('L',x1,yo,z1);
end;

begin {-main-}
  Th:=30;  Ph:=50;
  Init_Graph;
  Xmax:=GetMaxX div 2;
  Ymax:=GetMaxY div 2;
  SetActivePage(0);
  SetVisualPage(0);clearViewPort;
  XYZ;
  curve;
  cube(15,0,0,17,2,2,15);
  Ipage:=1;
  SetActivePage(Ipage);
   for i:=1 to 50 do
   begin
      Ph:=Ph+0.2;                          {ภาพหมุนโดย เพิ่มค่ามุม Ph และ Th}
      Th:=Th+2;
      ClearViewPort;XYZ;curve; cube(15,0,0,17,2,2,15);
      SetVisualPage(Ipage);
      if Ipage=0 then Ipage:=1 else Ipage:=0;
      SetActivePage(Ipage);
    end;
   ch:=readkey;
end.
```



รูปที่ 14 แสดงผลที่ได้จากตัวอย่างโปรแกรมระบบ I จะหมุนรอบกล่องไม่ได้ หมุนได้แต่รอบจุดกำเนิดเท่านั้น

ตัวอย่างโปรแกรม Viewing syste m IV

ในระบบ IV ใช้ผลคูณทางแมตริกซ์ 3 ตัว คือ

$$T=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ -C_x & -C_y & -C_z & 1\end{bmatrix}, B=\begin{bmatrix}U_x & V_x & N_x & 0\\ U_y & V_y & N_y & 0\\ U_z & V_z & N_z & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

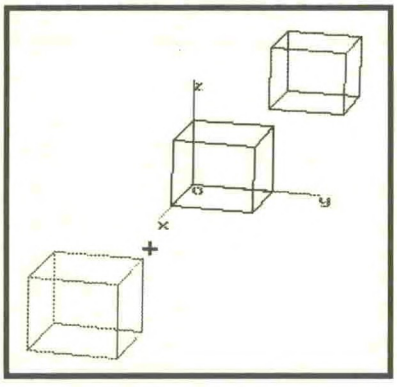
โดย U,V,N เป็น unit vector ตั้งฉากกันบนจุด C และเวกเตอร์ V หาได้จาก

$$V = Vup-(Vup.N)N \text{ และ } U=NxV$$

ทำ normalized อีกครั้งเพื่อให้ค่า z สุดท้ายอยู่ในช่วง [0, 1] โดยใช้แมตริกซ์

$$T_{pers}=\begin{bmatrix}d/h & 0 & 0 & 0\\ 0 & d/h & 0 & 0\\ 0 & 0 & f/(f-d) & 1\\ 0 & 0 & -df/(f-d) & 0\end{bmatrix}$$

$$T_{view} = TBT_{pers}$$



รูปที่ 15
แสดงผลที่ได้จากตัวอย่างโปรแกรมระบบ IV หมุนรอบจุด '+'

ถ่ายทอดออกเป็นเป็นโปรแกรมได้ตามตัวอย่างโปรแกรมที่ 2

เอกสารอ้างอิง

[1] Alan Watt(1993) 3D Comp uter Graphics,Second edition,Addison-Wesley.
[2] Foley,J.D.,A. Van Dam, Feiner,S.K.,Hughes J.F. and Phillips R.L.,(1994) Introduction to computer graphics, Addison-Wesley Publishing Company, Inc.

โปรแกรมที่ 2

```
program Example_2_Draw_Cube_with_Viewing_system_IV;
{== Somchart R.SORAKARN,
     Dept. of Comp. Science,
       KMITNB, BKK 10800.===}
uses crt,graph;
var                    ch2 : char;
      i,j,k,Ipage,X_max,Y_max,sx,sy,MM: integer;
    cx,cy,cz,ux,uy,uz,vx,vy,vz,nx,ny,nz,vpx,vpy,vpz :real;
              Tview:array[1..4,1..4] of real;
                   wx,wy,wz:real;
                 w,K1,K2,d,h:real;

procedure Per(L:char;x,y,z:real);
var   xs,ys,zs:real; col:integer;
     xx:array[1..4] of real;

     function cal(j:integer):real;
      var tmp:real;
      begin
        tmp:=0;for i:=1 to 4 do tmp:=tmp+xx[i]*Tview[i,j];
        cal:=tmp;
      end;
begin
    xx[1]:=x; xx[2]:=y; xx[3]:=z;  xx[4]:=1;
     w:=cal(4);
    zs:=cal(3)/w;  {0<z<1}                          }
    ys:=cal(2)/w;  {-1<ys<1}                        } ผลการทำ normalized
    xs:=cal(1)/w;  {-1<xs<1}                        }
    if (zs>0) and (zs<1) then
    begin
     sx:=trunc(1.2*(X_max + xs*MM));   {screen 640x348,MM=174}
     sy:=trunc(Y_max - ys*MM);
      case upcase(L) of
        'L': LineTo(sx,sy);
        'M': MoveTo(sx,sy);
      end; {case}
    end;
end;  {per}
procedure XYZ;
var Ox,Oy:integer;
begin
  setcolor(3);
  per('M',0,0,0);    OutTextXY(sx,sy,'o');Ox:=sx;Oy:=sy;
  per('L',0,0,1.6);  OutTextXY(sx,sy,'z');MoveTo(Ox,Oy);
  per('L',0,1.6,0);  OutTextXY(sx,sy,'y');MoveTo(Ox,Oy);
  per('L',1.6,0,0);  OutTextXY(sx,sy,'x');
end;
procedure Init_Graph;
var gd,gm:integer;
begin
   gm:=1; gd:=9; initGraph(gd,gm,'');
end;
```

→

โปรแกรมที่ 2 (ต่อ)

```
procedure Cube(xo,yo,zo,x1,y1,z1,co:integer);
begin
  setcolor(co);
  per('M',x1,yo,zo);per('L',x1,y1,zo);per('L',x1,y1,z1);
                per('L',x1,yo,z1);per('L',x1,yo,zo);
  per('M',xo,yo,zo);per('L',xo,y1,zo);per('L',xo,y1,z1);
                per('L',xo,yo,z1);per('L',xo,yo,zo);
  per('M',xo,y1,zo);per('L',x1,y1,zo);
  per('M',xo,y1,z1);per('L',x1,y1,z1);
  per('M',xo,yo,zo);per('L',x1,yo,zo);
  per('M',xo,yo,z1);per('L',x1,yo,z1);
end;
procedure Mat;
var sum:real;
   T_,B_,Mp_,tmp1,tmp2,Tper:array[1..4,1..4] of real;
begin
   T_[1,1]:=1;  T_[1,2]:=0;   T_[1,3]:=0;   T_[1,4]:=0;
   T_[2,1]:=0;  T_[2,2]:=1;   T_[2,3]:=0;   T_[2,4]:=0;
   T_[3,1]:=0;  T_[3,2]:=0;   T_[3,3]:=1;   T_[3,4]:=0;
   T_[4,1]:=-cx; T_[4,2]:=-cy; T_[4,3]:=-cz; T_[4,4]:=1;
   B_[1,1]:=ux;  B_[1,2]:=vx;  B_[1,3]:=nx; B_[1,4]:=0;
   B_[2,1]:=uy;  B_[2,2]:=vy;  B_[2,3]:=ny; B_[2,4]:=0;
   B_[3,1]:=uz;  B_[3,2]:=vz;  B_[3,3]:=nz; B_[3,4]:=0;
   B_[4,1]:=0;  B_[4,2]:=0;   B_[4,3]:=0;  B_[4,4]:=1;
 {Tpers, Homogeneous screen coordinate system}
   Tper[1,1]:=d/h; Tper[1,2]:=0;  Tper[1,3]:=0;          Tper[1,4]:=0;
   Tper[2,1]:=0;  Tper[2,2]:=d/h; Tper[2,3]:=0;          Tper[2,4]:=0;
   Tper[3,1]:=0;  Tper[3,2]:=0;  Tper[3,3]:=k2/(k2-k1);   Tper[3,4]:=1;
   Tper[4,1]:=0;  Tper[4,2]:=0;  Tper[4,3]:=-k2*k1/(k2-k1); Tper[4,4]:=0;
   for i:=1 to 4 do
     for j:=1 to 4 do
       begin
         sum:=0;
          for k:=1 to 4 do
            sum:=sum+T_[i,k]*B_[k,j];
               Tmp1[i,j]:=sum;
       end;
   for i:=1 to 4 do
     for j:=1 to 4 do
       begin
         sum:=0;
          for k:=1 to 4 do
            sum:=sum+Tmp1[i,k]*Tper[k,j];
           Tview[i,j]:=sum;
       end;
end; {Mat}
procedure Unt(var x,y,z:real);
var t:real;
begin
   t:=sqrt(x*x+y*y+z*z); x:=x/t; y:=y/t; z:=z/t;
end;
procedure nnn;
var vdn:real;  {v dot n}
begin
  nx:=wx-cx;  ny:=wy-cy; nz:=wz-cz;
  Unt(nx,ny,nz);
  vdn:=vpx*nx + vpy*ny + vpz*nz;  {v'. N}
  vx:=vpx-vdn*nx;                                {หาเวกเตอร์ขึ้น v' - (v'. N)N}
  vy:=vpy-vdn*ny;
  vz:=vpz-vdn*nz;
  Unt(vx,vy,vz);
  ux:=(ny*vz-nz*vy);           {u=nxv left-handed coordinate system}
  uy:=(nz*vx-nx*vz);
  uz:=(nx*vy-ny*vx);
end;

procedure display_window;
var st1,st2,st3:string[20];
  function RD(x:integer):Integer;  {round (x*1.20)}
    begin  RD:=round(x*1.2); end;
begin
```

→

โปรแกรมที่ 2 (ต่อ)

```
    setcolor(15);
    MoveTo(RD(X_max-MM),Y_max-MM);
    LineTo(RD(X_max+MM),Y_max-MM);
    LineTo(RD(X_max+MM),Y_max+MM);
    LineTo(RD(X_max-MM),Y_max+MM);
    LineTo(RD(X_max-MM),Y_max-MM);
        setcolor(12);  {display viewing point}
        per('M',wx,wy,wz);
        setcolor(13); OutTextXY(sx,sy,'+');
end;
var ii:integer;
begin {-main-}
   nosound;
   D:=5;  h:=0.4;       {D,h : control the projected image on screen}
   k1:=d; k2:=200;      {D,k1,k2 : View,Near,Far plane}
   cx:=45.0; cy:=45.5; cz:=15.5;    {cx,xy,cz = camera position}
   wx:=4.0; wy:=0.5; wz:=0.5;       {viewing point}
   vpx:=0.0; vpy:=0.0;  vpz:=1.0;   {v'=(0,0,1) up vector}
 Init_Graph;
 X_max:=280;
 Y_max:=GetMaxY div 2;
 MM:=Y_max;
 nnn;
 Mat;
 SetActivePage(0);
 SetVisualPage(0);clearViewPort;
 XYZ;
 Cube(0,0,0,1,1,1,7);
 Cube(6,0,0,7,1,1,2);
 display_window;
 Cube(-5,0,0,-4,1,1,6);
 ch2:=readkey;
 Ipage:=1; SetActivePage(Ipage);
  for ii:=0 to 40 do
  begin
    ClearViewPort;
     cy:=cy-2;                                  {เลื่อนตำแหน่งกล้องทางแกน y}
     nnn;
     Mat;
      XYZ;
     Cube(0,0,0,1,1,1,7);
     Cube(6,0,0,7,1,1,2);
     Cube(-5,0,0,-4,1,1,6);
     display_window;
     SetVisualPage(Ipage);
     if Ipage=0 then Ipage:=1 else Ipage:=0;
     SetActivePage(Ipage);
  end;
 ch2:=readkey;
end.
```